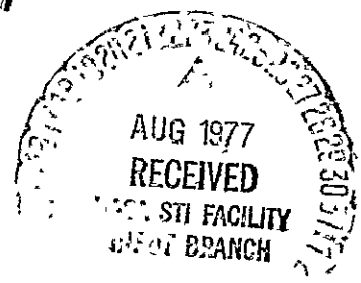
X-626-77-163

TMX-71366

GENERAL PURPOSE FILM PLOTTING SYSTEM

CHRISTINE McQUILLAN

JUNE 1977



GSFC

GODDARD SPACE FLIGHT CENTER
GREENBELT, MARYLAND

GENERAL PURPOSE FILM PLOTTING SYSTEM

Christine McQuillan

Laboratory for Planetary Atmospheres

Code 626

June 1977

TABLE OF CONTENTS

SECTION 1.0 PROGRAM DESCRIPTION

The General Purpose Film Plotting System (GPFPS) is a highly flexible, easy to use plot program designed to handle a majority of the data tape formats presently available under OS/360. The original version of this program was written by G. Masaki (Wolf Research and Development Corporation) in 1971 for G. Mason (Goddard Space Flight Center). The program was subsequently modified and updated by G. Mason and the author. This documentation reflects the final update of the system. The computer program is available in the GSFC computer program library.

The convenience of GPFPS is due to the fact that the user merely describes, in a prescribed manner, the format of his data set and the type of data plots he desires. GPFPS processes the input data according to the given specifications. The output is generated on a tape, which yields data plots when processed by the selected plotter (SD-4060, Calcomp, or Gerber). A summary of each job is produced on the printer.

The flexibility of GPFPS is reflected in the wide variety of plot formats available. Grids may be linear, semilogarithmic, or full logarithmic. Points may be plotted, or lines or points with lines. A single grid may be placed on each frame or up to three grids may be stacked, one above another. If there is one grid per frame, up to ten different ordinate variables may be plotted against the same abscissa.

Great flexibility is allowed in the selection of data to be plotted from the input data set. Every record for which the abscissa and ordinate variables are within the specified ranges may be used. Optionally, every Ith such record may be used. Another method is to plot every such record for which a third variable is within a specified

being initiated whenever a specified variable exceeds the limits set for it.

The operation of the General Purpose Film Plotting System is firmly based on the General I/O package (FTIO and DAIO) and the plot segment of the Wolf Plot and Contour Package (WPCP). An understanding of these software packages is assumed in this user's guide.

The major change made by the author in GPFPS is the use of the final version of the Wolf Plotting and Contouring Package. This change has been tested in all elementary modes of operation; namely, single plots, multiple plots, multiple grids (See Section 3.3). All other program options, such as data interpolation and direct access data organization, have been neither changed nor tested by the author.

REFERENCES

1. Parker, Thomas ., FTIO-FORTRAN I/O Package, March 1976, Computer Sciences Corporation, CSC/TM-76/6069.

2. Dickman, Charles I., IBM System/360 General I/O Package, March 1973, GSFC, Laboratory for High Energy Astrophysics.

3. Masaki, Geoffrey T., The Wolf Plotting and Contouring Package, January 1972 (revised by Computer Sciences Corporation, April 1976), GSFC Library #A00227.

SECTION 2.0 INPUT PROGRAM REQUIREMENTS

2.1 Input Data Set

The input data set is read in by the General I/O Package on FORTRAN unit 20. The entire current record is stored in core, in the COMMON block READ$$. Two forms of data set organization are permitted: physical sequential and direct access.

Direct access organization is the more restricted form. It requires a direct access unit, such as a disk, and is handled by a set of subroutines designated DAIO in the Generalized IO Package. The record format must be fixed; each data record is of the same length.

A data set with physical sequential organization may reside on either tape or disk. This type of input is handled by FTIO. In this case the record format may be fixed or variable, blocked or unblocked, spanned or unspanned. Variable format records must conform to the following specifications: a fixed length segment, followed by repeating variable segments of fixed size. In order for the system to determine the length of these records, either the number of variable segments must be indicated in the fixed segment of each record or else this number must be fixed for all records.

Read errors are handled by the Generalized IO Package. A description of the error is contained in the COMMON block FERMSG, and is output on the printer log of the job.

EXAMPLE:

```
**MODULE GPFPS**I/O ERROR ON RECORD 10
```

A count is kept of the number of unreadable records encountered in

processing data; it is listed on printer summary for each frame of data.

2.2 Control Input Specifications

The control data set is normally input on cards on FORTRAN unit 5. However, any input in 80-byte records in a single file on unit 5 is acceptable. Most control specifications are assigned default values in a BLOCK DATA subroutine. Only those which are to be overridden and those which are not assigned default values are required in the control data set.

Input control cards are all listed on the printer summary for each job. Errors are noted on each card, with a pointer to the approximate location, and a number indicating the incorrect control variable.

EXAMPLE:

```
**MODULE READCC**...ERROR IN CONTROL CARD...SCAN POINTER = 12.
     VARIABLE IS TYPE 10
```

A summary of all control specifications, set either by the user or by default, is also given on the printer log. It reflects the setting of all parameters for program execution. The program terminates at this point if any errors were encountered in the control data set.

EXAMPLE:

```
**MODULE READCC**EXECUTION TERMINATING DUE TO ERRORS
     IN CONTROL CARD INPUT
```

The control variables which may be specified in the control data set are discussed individually in section 2.3. Through these variables the user describes both his input data set and the type of output desired.

The remainder of this section discusses the general specifications for the control input data set.

1) The control data set consists of a number of control specifications.

2) The control specifications are free-form.

   a. All columns of the input card may be used.

   b. The end of data on a card is denoted by a dollar sign ($).

   c. Each specification must begin and end on the same card.

   d. More than one specification may be given per card. The items must be separated by a semicolon (;).

   e. A specification which has more than one associated value has these values separated by commas (,).

   f. A subscripted specification, each of which has more than one associated value, has these sets of values separated by colons (:).

   g. Comment cards, flagged with an asterisk (*) in column 1, may be placed in the control data set as desired.

3) Construction of control specifications:

   KEYWORD (I) = A, B, C, D;

   a. KEYWORD is one of the control variables in Table I. It is a character string, and must be spelled as indicated.

   b. I is an optional subscript used with some control variables.

   c. A, B, C, D denotes an optional list of items, either numbers or character strings enclosed in single quotation marks. This list serves a variety of purposes depending on the

associated control variable. For example, it may specify an abscissa label (TITLEX = 'ALTITUDE';), or the range and number of increments used on the abscissa (RANGEX = 0., 100., 4;).

4) Alternate constructions for subscripted control specifications:

a. If the list associated with each subscripted keyword consists of a single item, then these items may be specified as a single list with items separated by commas, as long as the subscripts are consecutive.

EXAMPLE: If KEYWORD (I) = A, KEYWORD (I+1) = B, KEYWORD (I+2) = C, and I = 3, then the following two constructions are equivalent:

KEYWORD (3) = A; KEYWORD (4) = B; KEYWORD (5) = C; $

KEYWORD (3) = A, B, C; $

b. If the list associated with each keyword consists of more than one item, then the sets of values may be placed in a list, with the values separated by commas and the sets separated by colons.

EXAMPLE: If KEYWORD (I) = A_I, B_I, C_I, KEYWORD (I+1) = A_{I+1}, B_{I+1}, C_{I+1} and I = 3, then the following two constructions are equivalent.

KEYWORD (3) = A_3, B_3, C_3; KEYWORD (4) = A_4, B_4, C_4; $

KEYWORD (3) = A_3, B_3, C_3: A_4, B_4, C_4; $

c. If, in either case, the first specified subscript is 1, then it may be omitted:

KEYWORD = A, B, C; $

KEYWORD = A_1, B_1, C_1: A_2, B_2, C_2; $

5) If a control specification is defined more than once, the latest definition overrides all previous ones.

2.3 Glossary of Control Specifications

This section discusses each control specification separately, and is summarized in Table I. The items are numbered to correspond to that table. These numbers are also used to flag errors on input control cards.

1) RECFM specifies the type of records contained on the input tape. It is coded as follows:

1 means fixed length, and is the default.

2 means variable length records of the type described in section 2.1.1.

EXAMPLE: RECFM = 1;

2) RECSIZ specifies the maximum record length in bytes. There is no default value supplied for this keyword, which must be specified by the user.

EXAMPLE: RECSIZ = 496;

3,4,5) These keywords specify the structure of a variable record. They are not used with fixed records, and are ignored if specified.

FIXSIZ = number of bytes in the fixed segment.

VARSIZ = number of bytes in the variable segment.

COUNT = number of variable segments per record. It may be a constant for all records, or a variable, in which case its value must be stored in the fixed segment of each record. This keyword is specified in the same manner as XD (see item 6).

If the record length is greater than FIXSIZ + COUNT*VARSIZ, then the part in excess is ignored.

EXAMPLE: Records are 22 bytes long, with a 7-byte fixed segment and 5 variable segments of 3 bytes each. The number of variable segments is therefore a constant, and is specified as shown below.

FIXSIZ = 7; VARSIZ = 3; COUNT = 5, 0, 4;

6) XD defines the abscissa variable in terms of the location within the record and its length (both in bytes) and type. Its location is specified relative to the start of the record; if the records are fixed length. If the records are of variable length, the location may be relative to either the start of the record, or of each segment, according to the value of another keyword (see item 8). The data type is indicated by the following code:

1 means integer

2 means real

3 means character

4 means that the value given as the starting location is to be used. In this case the indicated length is ignored.

EXAMPLE: To use as the abscissa value the seventh halfword integer in the record:

XD = 13, 2, 1;

EXAMPLE: To use ten as the abscissa value:

XD = 10, 0, 4;

7) YD(I) defines the ordinate variables in the same manner as XD defines the abscissa. The maximum value of the subscript is ten.

EXAMPLE: To use as the first ordinate the fifth single-precision word

in the record; and as the second, the character contained in the eleventh byte:

YD = 17, 4, 2: 11, 1, 3;

8) DATALOC specifies, for variable length records, the location of the abscissa and ordinate variables within the record. It is coded as follows:

1 means that all values are in the fixed segment. This is the default.

2 means that all values are in the variable segment.

3 means that the abscissa is located in the fixed segment, and the ordinate(s) are in the variable segment.

EXAMPLE: DATALOC = 3;

9,10) It may not be desirable to plot all of the data that meet the selection criteria. XFREQ and YFREQ(I) are used to specify the particular subset to be plotted for the Ith ordinate.

a. XFREQ = N_1, N_2, N;

Every Nth point, starting at N_1 and ending at N_2, is plotted for each ordinate.

EXAMPLE: XFREQ = 1, 11, 2; plots points 1, 3, 5, 7, 9, 11 for each ordinate.

b. YFREQ (L) = N_1, N_2, N;

Every Nth point, starting at N_1 and ending at N_2, is plotted for the Lth ordinate. All points are plotted for all other ordinates.

EXAMPLE: YFREQ (2) = 2, 11, 2; plots points 2, 4, 6, 8, 10 for the

second ordinate. All points are plotted for all other ordinates.

c. XFREQ = N_1, N_2, N; YFREQ (L) = M_1, M_2, M;

An even more restricted subset of points is plotted for the Lth ordinate. A subset is created containing every Nth point, starting at N_1 and ending at N_2. From this subset another is generated containing every Mth point, starting at M_1 and ending at M_2. This second subset is plotted for the Lth ordinate. For all other ordinates every Nth point, starting at N_1 and ending at N_2, is plotted.

EXAMPLE: XFREQ = 1, 11, 2; YFREQ (2) = 2, 11, 2: 2, 6, 4;

Points 3, 7, 11 are plotted for the second ordinate.

Points 3 and 11 are plotted for the third ordinate.

Points 1, 3, 5, 7, 9, 11 are plotted for all other ordinates.

11) MODE (I) specifies settings for the various plotter options available from the plot segment of the Wolf Plot and Contour Package. These are described in Table II.

12) The framebreak variable is used to reduce the amount of data plotted per frame. Instead of one frame, a series of frames is generated in which one variable, not necessarily the abscissa or ordinate variable, is regularly incremented. This keyword specifies that variable in terms of its starting address within the record, its length, and type, in the same manner as XD (see item 6).

FRMBRK = A, B, C;

The absolute value of A is the starting address. The sign of A indicates, for variable-length records only, the location of the

framebreak variable within the record. If A is negative, it is in the variable segment; otherwise, in the fixed segment.

EXAMPLE: To use as the framebreak variable the seventh halfword integer in the variable segment of the record.

```
FRMBRK = -13, 2, 1;
```

13) FRMRNG is used to set limits on the framebreak variable. It is specified in terms of a lower and an upper limit for this variable, and the increment to be used between these.

```
FRMRNG  = A, B, C;
```

(B-A)/C frames are generated. For the Ith frame, only data from records in which the framebreak variable has a value between A + (I-1) * C and A + I * C will be plotted. A new frame is initiated, and the frame range incremented, as soon as a record is encountered in which the framebreak variable exceeds the current frame range. Note that monotonically increasing values are assumed for the framebreak variable; the data set is not rewound between output plot frames. Increasing values are assumed for the framebreak variable; the input data set is not rewound between output plot frames. If FRMRNG is not specified, a new frame is initialized whenever the value of the framebreak variable changes. When the framebreak occurs, the remainder of the record is used for the next frame.

EXAMPLE: To plot all data for which the framebreak variable is between 50 and 100, in increments of 5, thus generating ten separate frames.

```
FRMRNG = 50, 100, 5;
```

The framebreak variable may be identical with the abscissa variable XD. In this case the frame limits are also used as the abscissa limits, so that for the Ith frame the abscissa ranges from A + (I-1) * C to A + I * C. The number of intervals

on the grid between these limits is given by the third value of RANGEX (see item 14).

EXAMPLE: To increment the abscissa variable from -100 to +100, in 50-unit increments (on 4 separate frames), with 5 intervals per grid.

XD = 1, 4, 2; RANGEX = -100, 100, 5; $

FRMBRK = 1, 4, 2; FRMRNG = -100, 100, 50; $

A special option has been provided to handle a data set in which the abscissa variable increases monotonically in a periodic manner. This means there are multiple successive monotonically increasing sequences within the set of abscissa values. For example, in the data set X_1 to X_p there may be 3 sequences: X_1 to X_n, X_{n+1} to X_m, X_{m+1} to X_p; with $n \geq 1$, $m \geq n+1$, $p \geq m+1$ and $X_{n+1} < X_n$, $X_{m+1} < X_m$. Without using this special option all data would be plotted on the same frame (if the abscissa range is defined as the minimum and maximum values of the entire data set). Cyclic abscissa variation is permitted when the second value of the frame range keyword is set to a negative number. GPFPS will then use the abscissa range values for the framebreak range, and will start a new frame whenever the current value of the abscissa variable is less than its previous value.

EXAMPLE: Assume that the following XY pairs have been selected for plotting.

X: 1 2 3 4 2 3 1 5 8 3 5 7

Y: 1 2 3 4 5 6 7 8 9 10 11 12

Then the following specifications yield the indicated outputs:

RANGEX	FRMRNG	# Frames	# Points/Frame	Abscissa Range
1, 8, 2;	--	1	12	1 to 8
1, 4, 2;	--	1	8	1 to 4
1, 8, 2;	0, -1, 0;	4	4, 2, 3, 3	1 to 8
1, 4, 2;	0, -1, 0;	4	4, 2, 1, 1	1 to 4

14) RANGEX specifies the range of acceptable abscissa values. Values outside of this range are not plotted. In addition, it specifies the horizontal axis used for this frame.

RANGEX = A, B, C;

All plotted data will have values in the range A to B. The X-axis will span A to B in C intervals, if the scale is linear. If the scale is logarithmic, there will be C intervals per cycle. The keyword MODEX specifies the scale type (see item 21).

EXAMPLE: For a linear X-axis that is scaled from -100 to +100 in increments of 25 units.

RANGEX = -100, 100, 8;

Note that if the framebreak variable is the same as the abscissa value, then the framebreak limits for this frame are also used as the abscissa range. RANGEX (3) specifies the number of intervals used.

EXAMPLE: XD = 13, 2, 1; RANGEX = -100, 100, 8; $
FRMBRK = 13, 2, 1; FRMRNG = -50, 50, 20; $
This will produce 5 frames, with abscissas 20 units long starting at -50, -30, -10, 10, 30 units respectively. Each abscissa will be divided into 8 intervals.

15) RANGEY(I) specifies the acceptable range of values for the Ith ordinate in the same way that RANGEX does for the abscissa. The type of scale used is specified by the keyword MODEY(I) (see item 22). The values given for the first ordinate are used in drawing the grid itself. Each additional ordinate will have a separate vertical axis drawn to the left of the grid, thus reducing the space on the frame for the grid itself. There is only one exception: if all the ordinate scales are the same, then only one vertical axis will be drawn.

EXAMPLE: To use a scale from -1 to 1, with 8 intervals for the first ordinate; and from 10^4 to 10^6, with 2 intervals per cycle for the second (which is logarithmic)

```
RANGEY = -1., 1., 8: 1.E4, 1.E6, 2; MODEY(2) = 2;
```

16) LINE(I) specifies the particular combination of points and lines to be drawn for the Ith ordinate. It is coded as follows:

1 means that each data point is to be plotted with the symbol specified by the keyword SYMBOL (see item 18). This is the default.

2 means that line segments connect successive points.

3 means that each point is plotted, and successive points are connected by a line.

EXAMPLE: The first four ordinates are plotted with points only, and the fifth with line segments.

```
LINE(5) = 2;
```

17) STACK specifies the number of complete plot grids drawn on each

output frame. There are two options, with slightly different restrictions.

The first, which is the default, is one grid per frame, with the vertical axis equal in height to the frame height. In this case it is possible to plot up to ten ordinates on the same frame, as discussed under item 17. See first two sample outputs.

The second option is to draw up to three separate grids per frame. These will be placed one above the other on the frame, and will occupy equal areas. Only one ordinate is allowed per grid. This option is coded by number. If option 2 is specified (STACK = 2), then the number of grids drawn is the number of ordinates specified by YD. See third sample output.

EXAMPLE: To plot two ordinates on one grid, with ranges -1 to 1 and 10 to 16, respectively with 6 increments each. The ordinates are the second and third single-precision real words in the record.

YD = 5, 4, 2: 9, 4, 2; RANGEY = -1., 1., 6: 10., 16., 6;

EXAMPLE: The same two ordinates are to be plotted on separate grids

YD = 5, 4, 2,: 9, 4, 2; RANGEY = -1., 1., 6: 10., 16., 6; STACK = 2;

18) SYMBOL(I) specifies the character to be plotted for the Ith ordinate, if points are to be plotted (see item 16). The default for the Ith ordinate is the Ith letter of the alphabet. Each character is specified within single quotation marks.

EXAMPLE: * is to be used for the first ordinate, and X for the second.

SYMBOL = '*', 'X';

19, 20) TITLEX and TITLEY(I) specify the titles to be used in labeling the abscissa and Ith ordinate, respectively. The text, up to forty characters in length, must be specified between single quotation marks. The default for the abscissa is 'XD'; for the Ith ordinate, 'YD(I)'.

EXAMPLE: TITLEX = 'GREENWICH MEAN TIME'; TITLEY = 'ALTITUDE', 'MAGNETIC LONGITUDE', 'L';

21, 22) MODEX and MODEY(I) specify the type of scale to be used for the horizontal and Ith vertical axis, respectively. The default in each case is linear. These keywords are coded as follows:

1 means linear

2 means logarithmic, with base 10.

EXAMPLE: A log-log grid is desired.

MODEX = 2; MODEY = 2;

23) All records on the input data tape are normally considered in producing a plot. RECUSE specifies that a subset of these are used. It is given in terms of the first and last records to be considered, and the increment to be used between these.

EXAMPLE: Every fifth record between the first and the hundredth is to be used.

RECUSE = 1, 100, 5;

24) MAXFRM specifies the maximum number of frames to be plotted. The default value is 9999.

EXAMPLE: In a test run only 10 frames are needed to check the microfilm output.

MAXFRM = 10;

25) The selection variable is used to limit the type of data records considered for plotting. Only those records for which the selection variable is within a specified range are tested further to determine whether they are acceptable for plotting. The remainder are ignored. The selection variable is defined in terms of its starting address within the record, its length, and type, in the same manner as FRMBRK (see item 12).

SELECT = A, B, C;

The absolute value of A is the starting address. The sign of A indicates, for variable-length records only, the location of the selection variable within the record. If A is negative, it is in the variable segment; otherwise, in the fixed segment.

EXAMPLE: To use as the selection variable the character value contained in the eleventh byte of the record.

SELECT = 11, 1, 3;

26) SELRNG is used to set limits on the selection variable. If it is omitted, any change in the value of the selection variable results in a new frame. This keyword specifies boundaries in the form

SELRNG = A, B, C;

where the meanings of the items in the list depend upon the type of boundaries being set. This is determined by the sign of C, the third item.

a. If C = 0, no incrementing is performed. The only selection range considered is A to B.

b. If C > 0, the range considered for the selection variable are A to B, A + C to B + C, A + 2C to B + 2C, and so on.

c. If C < 0, the absolute value of C is used to obtain the selection ranges A to A + C, A + C to A + 2C, A + 2C to A + 3C, and so on. Incrementing stops when the upper limit is greater than B.

EXAMPLE: To use as the selection variable the fifth REAL* 4 value in the record, in the ranges 0.0 to 1.0 and 1.0 to 2.0.

SELECT = 17, 4, 2; SELRNG = 0.0, 2.0, -1.0;

EXAMPLE: To use instead the ranges 0.0 to 1.0, 0.5 to 1.5, and 1.0 to 2.0.

SELECT = 17, 4, 2; SELRNG = 0.0, 1.0, 0.5;

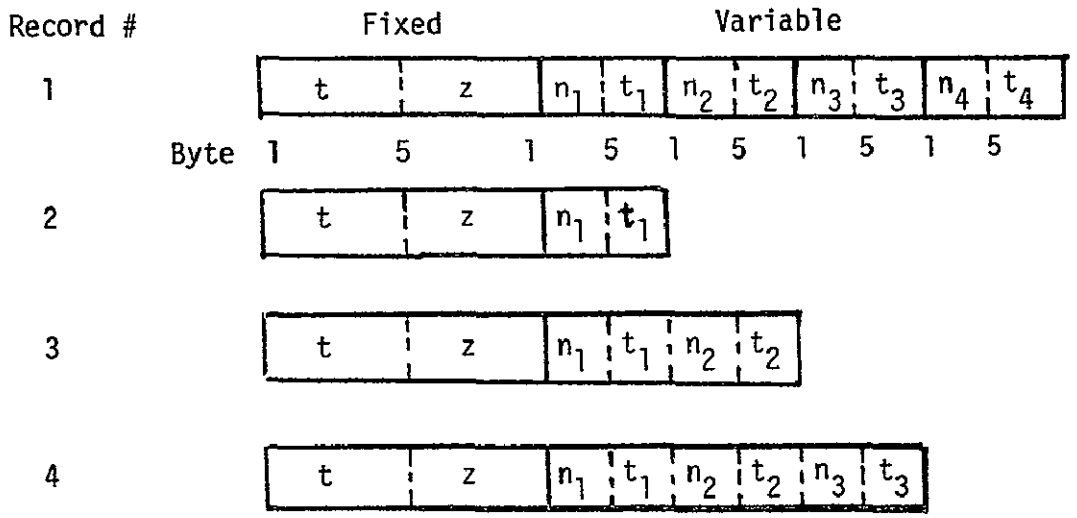
27, 28, 29) In the default mode, no interpolation is performed on the input data set. This is specified by setting the third value of INTX to zero (which is its default value). Interpolation is permitted on the input data set only if it consists of variable length records in which the abscissa variable is stored in the fixed segment and the ordinate variables in the variable segments. A Lagrangian interpolation polynomial of specified order is generated for each record. Abscissa values are obtained from it for all ordinate variables in that record. A new set of coefficients is determined for each record.

The keyword INTORD specifies the order of the interpolation polynomial. Its default value is 1, with 5 the maximum. The polynomial will, therefore, have N = INTORD + 1 coefficients, requiring N records for their determination. The coefficients for record J are generated from the set of records

J, J + 1	INTORD = 1
J - 1, J, J + 1, ..., J + INTORD - 1	INTORD > 1

For correct interpolation, these records must not have the same values for the variable to be interpolated.

The specification of the keywords INTX and INTY is best explained through an example. Consider a data set consisting of variable-length records as pictured below. All data is stored as single-precision real words.



Z is the spacecraft altitude at time t; these values are stored in the fixed segment of each record. N_i is a number density measured at a later time t_i; these are stored in the ith variable segment of the record. Interpolation is required if these densities are to be plotted as a function of altitude.

EXAMPLE: RECFM = 2; DATALOC = 3; $

DX = 5, 4, 2; YD = 1, 4, 2, $

INTX = 1, 4, 2; INTY = 5, 4, 2; INTORD = 2; $

A second-order Lagrangian interpolation polynomial is requested in this example. For the first record, the three coefficients are determined from records 1 to 3. The same records are used for

record 2. Record 3 uses records 2 through 4. Coefficients cannot be determined for record 4. Within a given record, the same coefficients are used to determine the altitudes corresponding to times t_i. This example assumes that the times t_i are absolute. If they are not, but are relative to the time t, then the third value of INTY should be set negative. The program will correct the times to absolute, using for interpolation the times $t + t_i$.

EXAMPLE: To indicate in the above example that the times t_i are elapsed times relative to time t.

```
INTY = 5, 4, -1;
```

Alternately the number densities n_i may be equally spaced in time throughout the record. The absolute times are then given by $t_i = t + C * (i-1)$, where C is the length of the time interval. In this case the times are not taken from the input data set, and may be absent from it. This type of correction is indicated by setting INTY to the data value C, as discussed under item 6.

EXAMPLE: To indicate 10 second intervals.

```
INTY = 10, 0, 4;
```

When interpolation is required, the program reads the entire input data set at once. The values specified by XD and INTX are stored in a temporary data set on FORTRAN unit 19. Both data sets are then rewound, and processing begins. Interpolation therefore requires two passes through the input data set.

30) RUNID is a string of eight characters used to identify, on the plot ID frame, the corresponding computer output. The default is the jobname on the computer run which generates the plots. This

string also appears on the first page of the print summary and above each output frame.

31) JOBTITLE is a string of up to forty characters which serves as a title for a single job. It is placed on the ID frame, along with RUNID. If the first character is a blank, the string also appears at the top of each plot. The default is a string of blanks.

EXAMPLE: JOBTITLE = ' UNCORRECTED EXPERIMENT DATA'

32) Values from the input data set may be used to identify output plot frames. A maximum of ten variables may be specified, in the same manner as the framebreak variable (see item 12). The corresponding values, obtained from the first record read for the current frame, will appear above the plot frame separated by slashes. The number of frame indentification values is independent of the number of ordinates; the variables may be identical with the abscissa, ordinate, framebreak, or selection variables.

EXAMPLE: To use as frame identification the second single precision word in the fixed segment of a variable length record, and the five-character string contained in bytes 1 to 5 of the current variable segment.

FRAMEID = 5, 4, 2: -1, 5, 3;

33) The organization of the input data set is indicated by the keyword DSORG. It is coded as follows:

1 = physical sequential (this is the default).

2 = direct access

This keyword insures that the proper access method is used by FTIO in acquiring records from the input device.

Table I

Summary of Control Specification Keywords

Item	Keyword	Default	Notes	Comments
1	RECFM	1		Input data tape format 1 = fixed 2 = variable
2	RECSIZE		6	Maximum record size (bytes)
3	FIXSIZ		1, 6	Fixed segment size (bytes)
4	VARSIZ		1, 6	Variable segment size (bytes)
5	COUNT		1, 2, 6	Number of variable segments
6	XD		2, 6	Abscissa variable (X)
7	YD		2, 4, 6	Ordinate variable (Y)
8	DATALOC	1	1	Distribution of data in record 1 = X and Y in fixed 2 = X and Y in variable 3 = X in fixed, Y in variable
9	XFREQ	all records	3	Plot frequency in terms of X for selected data
10	YFREQ	all records	3, 4	Plot frequency in terms of Y for selected data
11	MODE	see Table II		Settings for plotter options available from WPCP
12	FRMBRK	1 frame	2, 5	Framebreak variable, used to determine frame advance
13	FRMRNG	any change in framebreak variable	3	Limits on framebreak variable Also use for incremented abscissa variable and cyclic abscissa values (see Glossary
14	RANGEX		3, 6	Data range on horizontal axis
15	RANGEY		3, 4, 6	Data range on vertical axis
16	LINE	1	4	Method of plotting data 1 = points only 2 = line only 3 = line and points

Table I (continued)

Item	Keyword	Default	Notes	Comments
17	STACK	1		Number of grids per frame 1 = one full-sized grid 2 = up to three equal-sized grids
18	SYMBOL	Ith letter of alphabet	4	Plot symbol for Ith ordinate
19	TITLEX	'XD'		Abscissa title
20	TITLEY	'YD(1)' : 'YD(10)'	4	Ith Ordinate title
21	MODEX	1		Type of abscissa scale 1 = linear 2 = logarithmic
22	MODEY	1	4	Type of ordinate scale 1 = linear 2 = logarithmic
23	RECUSE	all records	3	Input records considered for plotting
24	MAXFRM	9999		Maximum number of frames plotted
25	SELECT	all points	2, 5	Selection variable, used to select data to be plotted
26	SELRNG	any change in selection variable	3	Limits on selection variable
27	INTX	no interpolation	1, 2	Data value in fixed segment used in interpolation (see Glossary)
28	INTY	no interpolation	1, 2	Data value in variable segment used in interpolation (see Glossary)
29	INTORD		1	Order of interpolation (5th is maximum)
30	RUNID	job ID		Job Identification

Table I (continued)

Item	Keyword	Default	Notes	Comments
31	JOBTITLE	blank		Job Title
32	FRAMEID	none	2, 4, 7	Frame identification from input record
33	DSORG			Input data set organization 1 = physical sequential 2 = direct access

---

Notes: These apply as indicated in the above table. Any exceptions are minor, and are fully explained in the glossary entry for each keyword.

Number	Explanation
1	This keyword applies to variable records only
2	This keyword is specified as KEYWORD = A, B, C; A = relative address of first byte B = length in bytes C = data type, coded as 1 = integer 2 = real 3 = character 4 = value A (B is ignored)
3	This keyword is specified as KEYWORD = A, B, C; A = minimum value B = maximum value C = number of increments between A and B (default is 1)
4	This keyword can be subscripted. The maximum value of the index is 10, in the unstacked mode; and 3, in the stacked mode.

Table I (continued)

Number	Explanation
5	This keyword is specified as indicated in NOTE 2. If the records are of variable length and A < 0, then the variable is located in the variable segment of the record. The absolute value of A then indicates the address of the first byte.
6	This keyword has no assigned default value. It must be defined by the user unless his records are fixed length and this keyword applies only to variable length records.
7	This keyword can be subscripted; the maximum value of the index is 10.

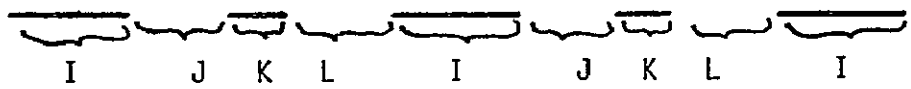
Table II

Plotting Options

I	MODE(I) default	Notes	Reference	Comments
1	0, 0, 0	1, 2		Printer plotter simulation
2	0, 0, 0	1, 2		SC-4020 plotter simulation
3	0, 0, 0	1, 2		Gerber plotter simulation
4	0, 0, 0	1, 2		Calcomp 30" drum simulation
5	0, 0, 0	1, 2		Calcomp 12" drum simulation
6	0, 0, 0	1, 2		Calcomp flatbed simulation
7	1, last, 1	2		SD-4060 plotter
8	1, 0, 0, 1		3.9	Affine transformation
9	T, '?'		4.2	Negative log error procedure
10	1		5.13	Set character size
11	1000	3, 4	8	Dashed line specifications
12	1	3	8	Number of lines plotted, side by side
13	1	3		Number of times a line is plotted over itself, giving a greater intensity
14	0			Printer plot frame size 1 = double page 2 = single page
15	6			FORTRAN output unit for printer plots
16	'E9.2.1'	5	5.8, 6.1	Abscissa labeling format
17	'E9.2.1'	3, 5	5.8, 6.1	Ith ordinate labeling format

Reference: Additional information may be obtained from the indicated section of the WPCP document.

Notes: These apply as indicated in the above table.

Number	Explanation
1	This keyword produces a simulation of the SD-4060 on the corresponding plotter.
2	The form of this keyword is KEYWORD = A, B, C; A = first frame simulated B = last frame simulated C = interval in frames (every Cth frame is processed)
3	This keyword can be subscripted. The maximum value of the index is 10, in the unstacked mode; and 3, in the stacked mode.
4	The form of this keyword is KEYWORD = A, where A is a 4-digit number IJKL. The digits specify, in units at 50 SD-4060 rasters, the length of dashes and the spaces between them, as follows: 
5	The form of this keyword is KEYWORD = 'A', where A is one of the following (w, d, t are integer values):

A	Equivalent FORTRAN format	Example
Fw.d		F10.4
Fw.d.t	tPFw.d	F10.4.0
Ew.d		E10.3
Ew.d.t	tPEw.d	E10.3.1
Iw		I2
Iw.t	tPIw	I2.0

SECTION 3.0 PROGRAM OUTPUT

The program output consists of a plot tape and a printer log. The requested data plots are obtained by processing the plot tape on the appropriate plotter. The printer log summarizes the computer job which produced the plot tape.

3.1 Printer Log

The front page of the printer log displays job identification information in block letters. This includes the ID of the programmer submitting the job, and the system-derived date and time at which it ran. All subsequent pages are numbered for convenience.

Page 1 lists the default values for all control keywords. These are numbered and listed as they appear in Table I. The only exception is the plotting options; the printer log reflects their listing in Table II.

The second page contains a numbered listing of the control data set. Errors are flagged on a card-by-card basis as indicated in Section 2.2.

Page 3 of the printer log lists the control data set as it is initialized for program execution. Its format is identical with that of page 1.

On subsequent pages a summary is generated for each data plot frame. This begins with the frame number and identification information derived from the keywords FRAMEID, JOBTITLE, and RUNID. Next is the number of points plotted for each ordinate variable, and a summary of rejected data points. Rejection occurs when the abscissa, ordinate, or selection

variable is out of the requested range. A point which fails more than one restriction is counted only once; the order of priority is SELRNG, XRANGE, YRANGE. The first and last records used and number of unreadable records are indicated. So that any change due to the use of the framebreak variable will be detected, the abscissa range is listed next. Finally, the plotters used and computer timing estimates are given. If no data points were plotted for the frame summarized on this page (i.e. no frame was actually output), the page ends with the message "NO PLOT THIS FRAME".

Error messages are output on the log as they occur. They will therefore precede the corresponding plot frame summary. Messages generated by GPFPS are prefixed with the name of the subroutine in which the error occurred. If no prefix appears, the message orginated in WPCP.

The end of the job is marked by a page containing the message "END OF GPFPS RUN".

It should be noted that if printer plots were requested on FORTRAN unit 6, each frame will follow its summary in the printer log.

3.2 PLOTTER OUTPUT

The first frame is an identification frame produced by the Wolf Plotting and Contouring Package. It contains the job title, run identification, and date on which the job was run.

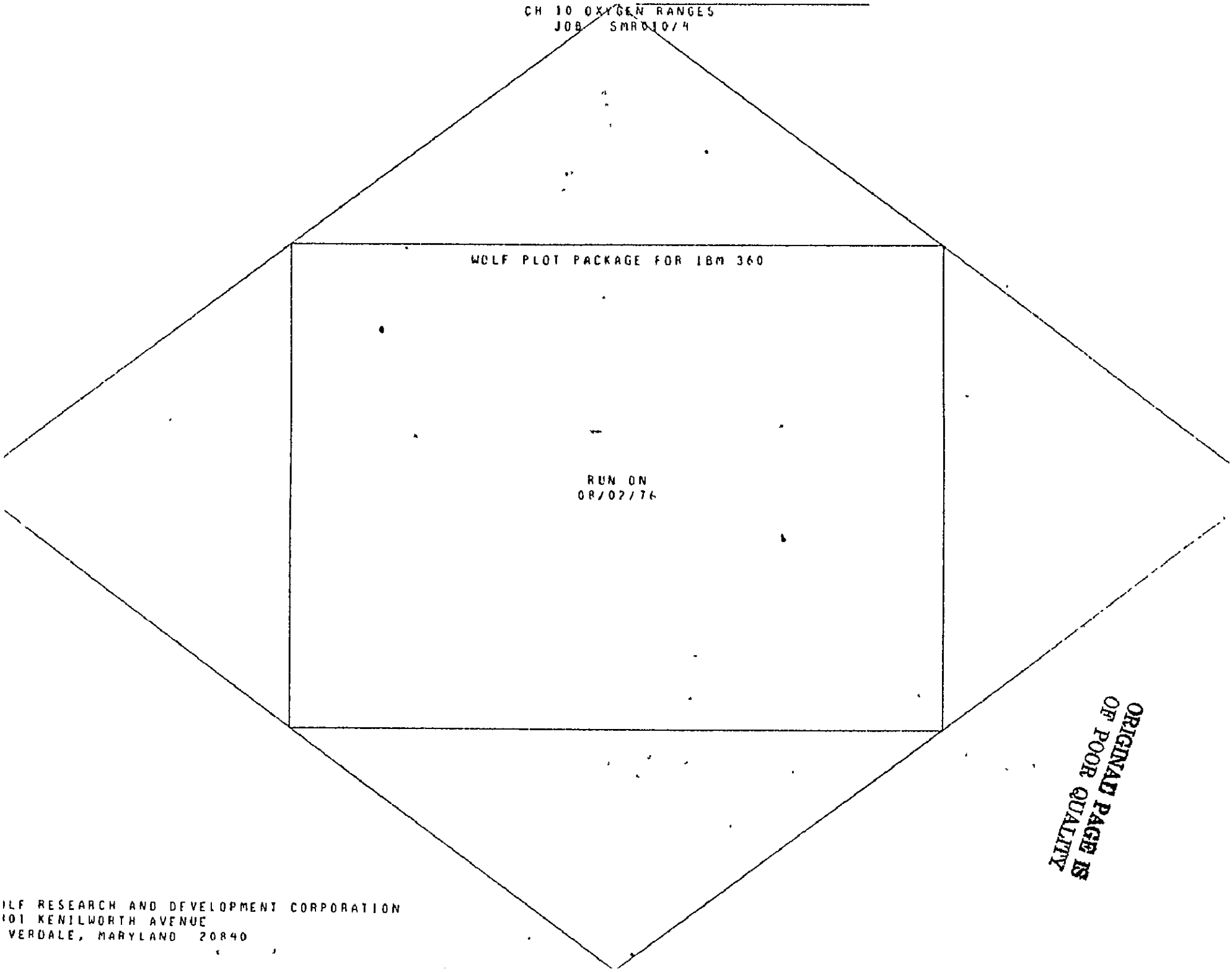
Each subsequent frame contains a data plot. The following information appears above each frame:

JOBTITLE RUNID FRAME TITLE	derived from the corresponding keyword
FRAME NUMBER	
JOB DATE & TIME	
REJECTED POINTS	6 numbers reflecting the number of data points < XRANGE, > XRANGE, < YRANGE, > YRANGE, < SELRNG, > SELRNG, respectively
NUMBER OF POINTS	for each ordinate variable. The plot symbol is also indicated.

The end of the job is flagged by a final identification frame. If the printer log indicates N date plots, the total number of frames to be output is N + 2.

3.3 SAMPLE SD4060 OUTPUT

Some examples of typical SD4060 plotter output are shown on the following pages. First is the identification frame. The second is the plot of a single variable on a single grid. The third and fourth both show two variables plotted, on the same and separate grids, respectively.

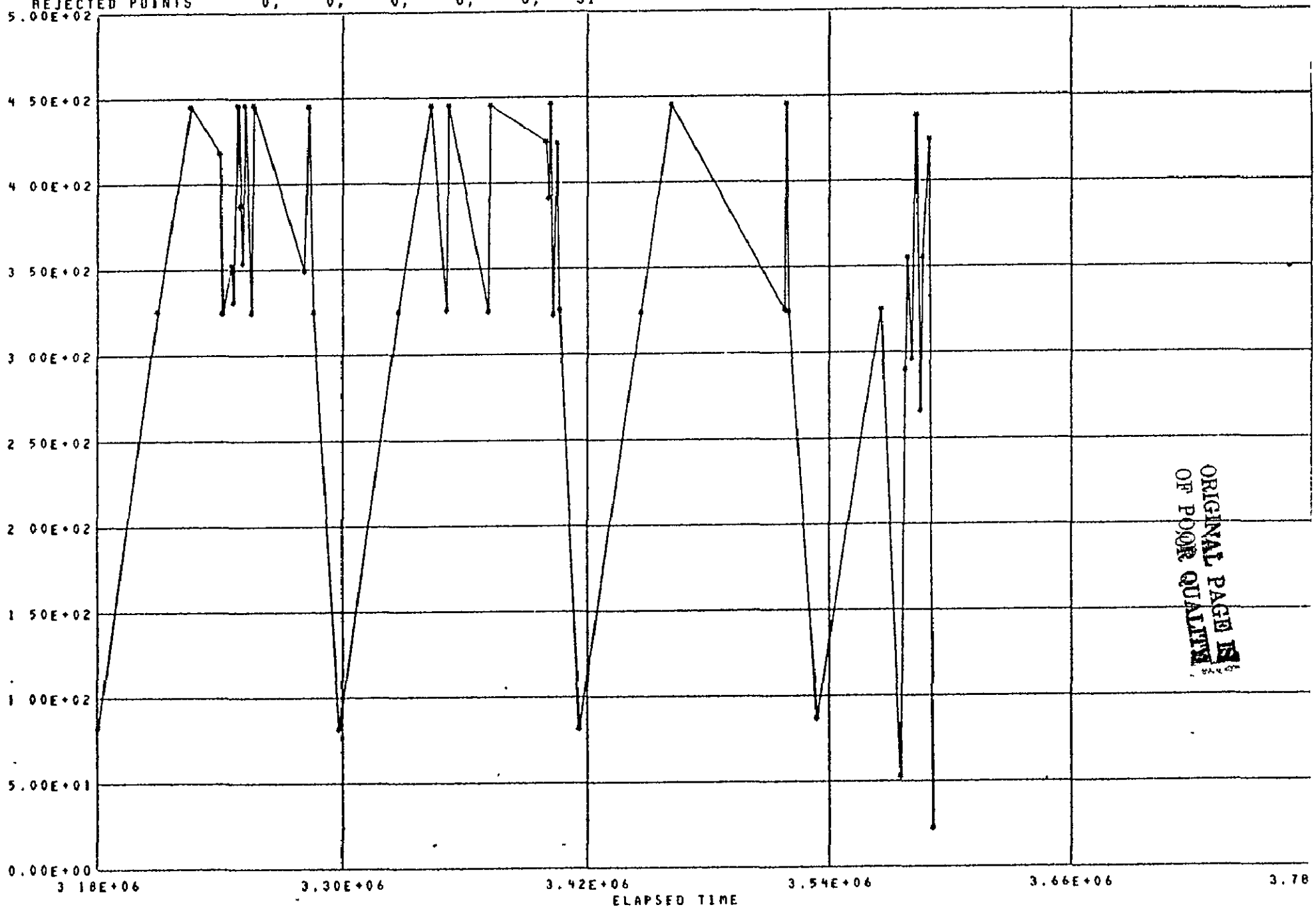


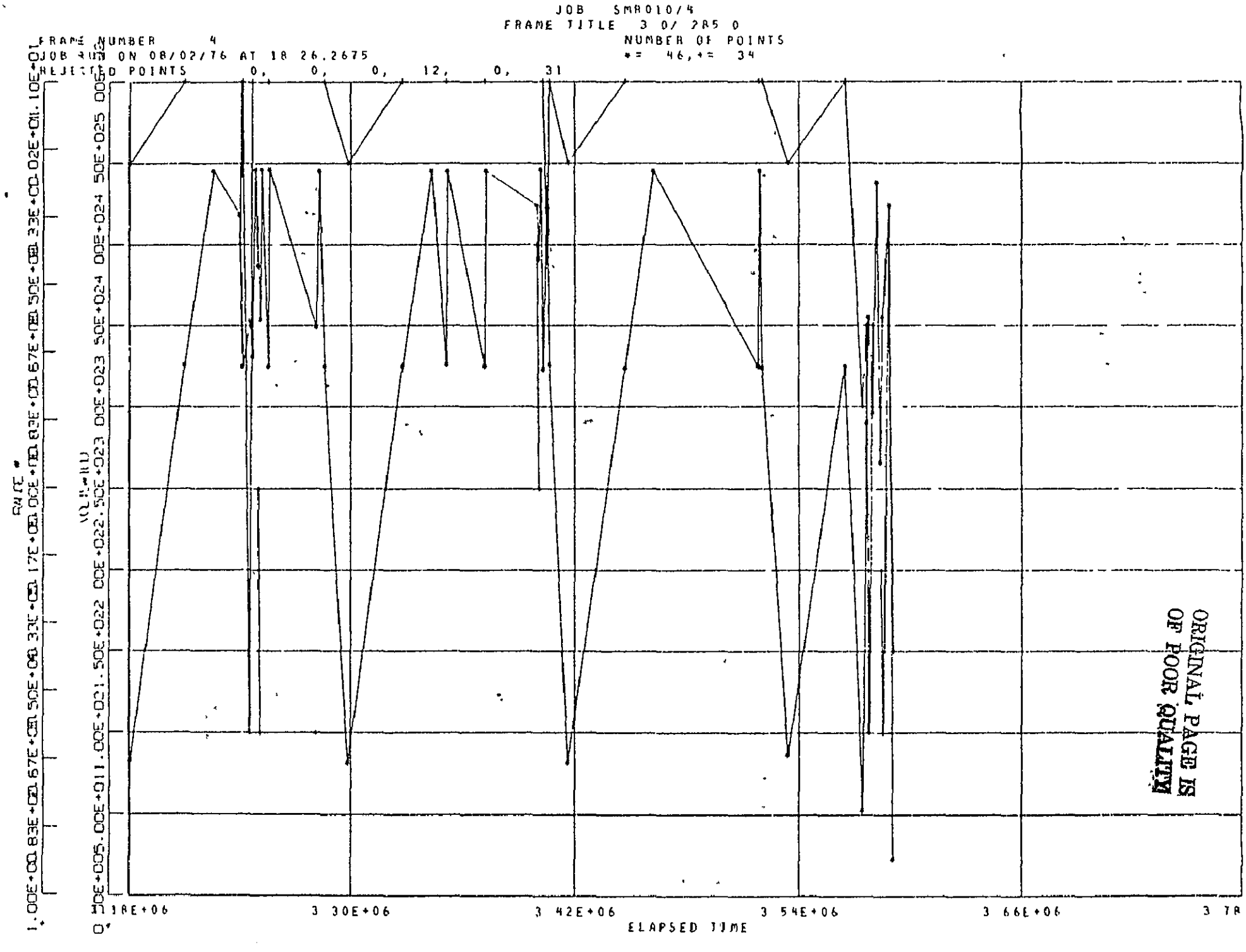
ILF RESEARCH AND DEVELOPMENT CORPORATION
101 KENILWORTH AVENUE
VERDALE, MARYLAND 20840

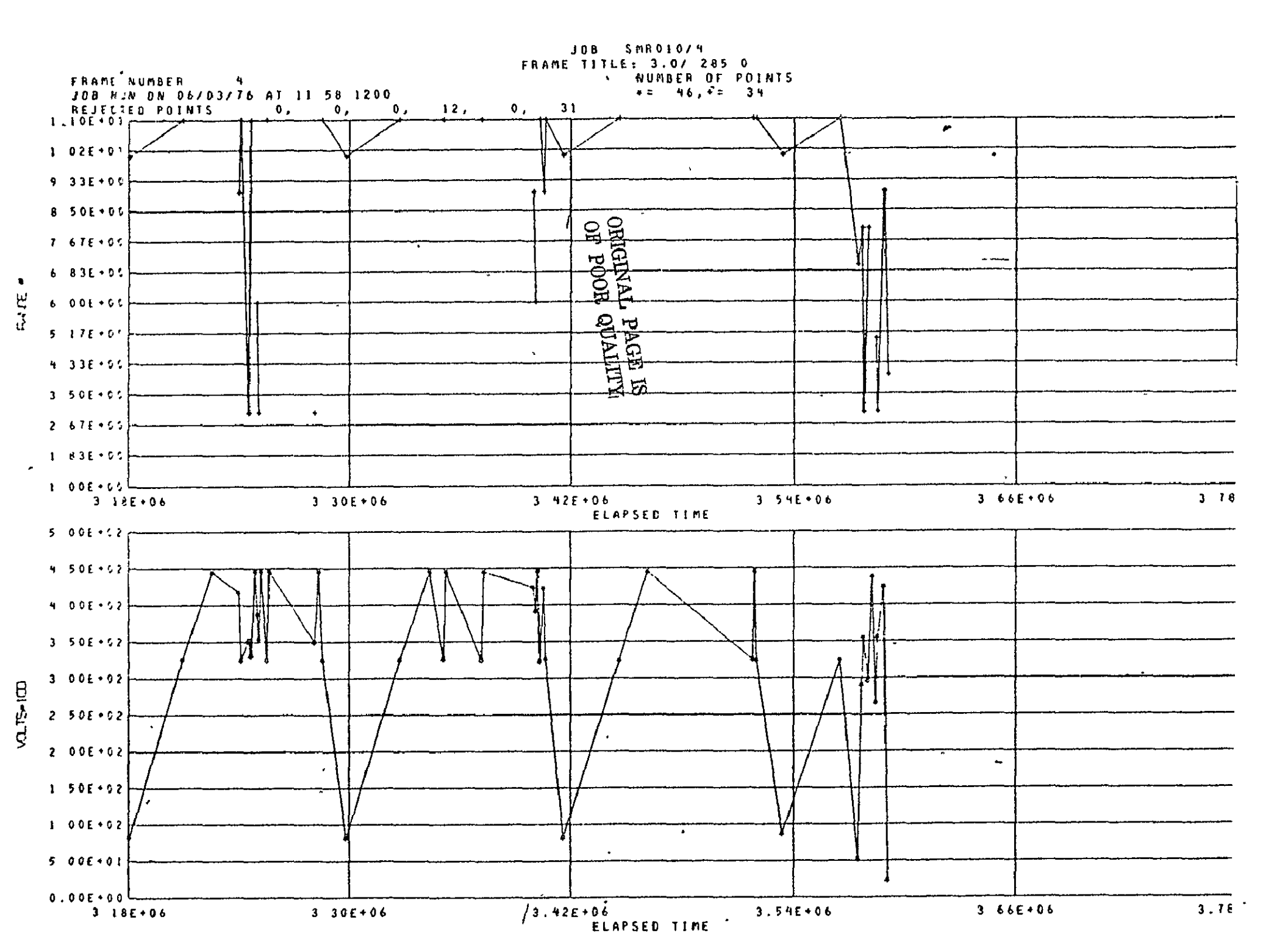
JOB SMR010/4
FRAME TITLE 3.0/ 285 0

FRAME NUMBER 4
JOB RUN ON 06/03/76 AT 10 21.4333
REJECTED POINTS 0, 0, 0, 0, 0, 31

NUMBER OF POINTS
*= 46




JOB SMR010/4
FRAME TITLE 3 0/ 285 0
NUMBER OF POINTS
*= 46,+= 34
FRAME NUMBER 4
REJECTED POINTS 0, 0, 0, 12, 0, 31
ELAPSED TIME
3 18E+06
3 30E+06
3 42E+06
3 54E+06
3 66E+06
ORIGINAL PAGE IS
OF POOR QUALITY


JOB SMR010/4
FRAME TITLE: 3.0/ 285 0
FRAME NUMBER 4
JOB RUN ON 06/03/76 AT 11 58 1200
NUMBER OF POINTS
*= 46, += 34
REJECTED POINTS 0, 0, 0, 12, 0, 31
1.10E+01
1.02E+01
9.33E+00
8.50E+00
7.67E+00
6.83E+00
6.00E+00
5.17E+00
4.33E+00
3.50E+00
2.67E+00
1.83E+00
1.00E+00
FRAME #
3.18E+06
3.30E+06
3.42E+06
3.54E+06
3.66E+06
3.78
ELAPSED TIME
ORIGINAL PAGE IS
OF POOR QUALITY
5.00E+02
4.50E+02
4.00E+02
3.50E+02
3.00E+02
2.50E+02
2.00E+02
1.50E+02
1.00E+02
5.00E+01
0.00E+00
VOLTS-100
3.18E+06
3.30E+06
3.42E+06
3.54E+06
3.66E+06
3.7E
ELAPSED TIME

SECTION 4.0 PROGRAM DOCUMENTATION

In this section the various subroutines that comprise the General Purpose Film Plotting System are documented. Programs contained in the Wolf Plot and Contour Package are not included. The following information when appropriate is provided for each subroutine:

I. Abstract: general description of the function of the program.

II. Background: author, date completed, source language, program size in bytes.

III. Requirements: calling sequence, calling routines, COMMON blocks, and non-system routines required.

IV. Method: the mathematical or logical procedure used.

V. Program flow: a general description of the program steps.

VI. Glossary of labels: constants, variables, and equivalences.

VII. Generated messages.

VIII. Constraints.

IX. Program listing.

All routines have been executed successfully on the IBM 360/91 under Release 21.8 of OS.

All COMMON blocks are fully described in section 4.16.

4.1 GPFPS

I. Abstract: main program GPFPS coordinates the activities of the General Purpose Film Plotting System.

II. Background:

A. Author: G. Masaki, G. Mason, C. McQuillan

B. Date completed: 7 May 1976

C. Source language: FORTRAN H

D. Program size: 88240 bytes

III. Requirements:

A. Calling sequence: not applicable.

B. Calling routines: not applicable.

C. COMMON blocks:

INPUT$, PLTGP$, READ$$, IO, FERMSG

D. Non-system routines:

CONCRD, VALUGP, GRIDGP, GETREC, LEGRGP, FRONT, REMTIM, CONMOV

The following are contained in the Wolf Plot and Contour Package:

NOW, EDIT, GRDNUM, DATE, IDFRME, PLOTST, VECTOR, PRINTR

IV. Method: see section V.

V. Program flow:

A. Initialize program variables for execution through use of input control data set and BLOCK DATA.

B. Check current record number. If it exceeds the maximum value requested, go to step G.

C. Read in next data record. For a read error, go to step E4k. For an end-of-file, go to step G.

D. Initialize program variables for the current record; in particular, determine the number of segments in this record. For fixed block records one variable segment, identical to the fixed segment, is assumed.

E. Perform the following steps for each segment:

1. Pick up the requested X and all Y values, either directly or through interpolation.
2. Determine if this X-value is in the set to be processed, according to the specification XFREQ. If it is greater than the maximum value requested, go to step G. If it is not in the specified set, go to step E5.
3. Initialize program variables for the current segment.
4. Perform the following steps for each ordinate variable.
   a. Determine if this y-value is in the set to be processed, according to the specification YFREQ(I).
      1) If it is greater than maximum, go to step E4j.
      2) If it is not in the specified set, go to step E4l.
   b. Check the framebreak variable:
      1) If X is cyclic:
         a) and the current value of X is not less than its previous value, go to step E4c.
         b) Otherwise, go to step E4b7.
      2) If there is none, go to step E4c.
      3) If it is contained in the fixed segment, this check is required once per record. Go to step E4c if it is not to be checked at this step.
      4) Determine its current value.
      5) If it is to be checked against its previous value:
         a) If there is none, store the value and go to

step E4c.

b) Otherwise, compare values. If equal, go to step E4c. Otherwise, go to step E4b7.

6) Otherwise check that its value is within the specified range. If not, go to step E4b7. Otherwise, go to step E4c.

7) Set parameters for the current plot. Go to step E4i.

c. Check the selection variable:

1) If there is none, go to step E4d.

2) If it is contained in the fixed segment, this check is required once per record. Go to step E4d if it is not to be checked at this step.

3) Determine its current value.

4) If it is to be checked against its previous value:

a) If there is none, store the value and go to step E4d.

b) Otherwise, compare values. If equal, go to step E4d. Otherwise, go to step E4c6.

5) Otherwise, check that its value is within the specified range. If not, go to step E4c6. Otherwise, go to step E4d.

6) Set proper counter for a point out of range. If the selection variable is in the fixed segment, go to step F. Otherwise, go to step E5.

d. Obtain the requested frame identification values.

e. Check the current x-value:

   1) If it is contained in the fixed segment, or in the variable segment without interpolation, this check is required once per record. Go to step E4f if it is not to be checked at this step.

   2) If it is within range, go to step E4f.

   3) Otherwise, set the proper counter for an off-scale point. Set parameters for the current plot and go to step E4g.

f. Check the value of the current y-variable:

   1) If it is within range, go to step E4g.

   2) Otherwise, set the proper counter for an off-scale point. Set parameters for the current plot.

g. Initialize the frame for the current plot and set up the grid, if this has not been done on a previous step.

h. If the current (X,Y) point is to be plotted:

   1) Count it, and place it in the plot buffer.

   2) If the buffer is not full, go to step E4l.

i. Otherwise, plot the accumulated data in the plot buffer for this ordinate variable, and reset the buffer.

j. End processing for the current (X,Y) point according to the following scheme:

Condition Encountered	Next Step
E4e1	E5
E4e3	E41
E4f2, E4a1	E5 F if all data requested by (YFREQ(I), I=1,NVAR) processed
E41	E5
E4b6	F

k. Flag a read error on the input data set. Go to step G, thus shipping the unreadable record.

l. End of loop over ordinate variables.

5. End of loop over record segments.

F. Update the record count. If it is within the specified range, go to step B.

G. Generate the summary printout for this frame, and output the frame on the requested plotters.

H. Reset, and increment as required, program variables.

I. Terminate execution when one of the following conditions occur:

1. End-of-file on the input data set.
2. Framebreak variable exceeds its upper limit, when such a limit has been specified.
3. Selection variable exceeds its upper limit, when this form of selection has been specified.
4. Number of frames output exceeds the specified maximum.

VI. Glossary of labels:

A. Constants

KMAX maximum dimension of XX and YY buffers

SLASH EBCDIC slash, used to separate frame titles

EFMT Wolf Plot and Contour Package representation of the FORTRAN format 1PE9.2, used for frame titles

IZERO binary zero, used to zero out arrays.

B. Variables

ICPU1 CPU time
IO1 IO time
} remaining at start of execution

ICPU2 CPU time
IO2 IO time
} remaining at start of frame

ICPU3 CPU time
IO3 IO time
} remaining at end of frame

ICPU4 CPU time
IO4 IO time
} required for this frame

ICPU5 CPU time
IO5 IO time
} required to this point

INTERP logical flag indicating if interpolation is requested

SINGLE logical flag indicating if only one frame is to be output

VARF logical flag indicating if the framebreak variable is in the variable segment of the record

VARS logical flag indicating if the selection variable is in the variable segment of the record

IBREAK displacement of framebreak variable relative to start of each variable segment

ISELCT displacement of selection variable relative to start of each variable segment

IFACTF integer flag indicating location of framebreak variable (1 = variable segment, 0 = fixed segment)

IFACTS integer flag indicating location of selection variable (1 = variable segment, 0 = fixed segment)

IFMAX length of the framebreak variable (must not exceed 8 bytes)

ISMAX length of the selection variable (must not exceed 8 bytes)

FRMSTR logical flag indicating if a frame has been started

SETRNG logical flag indicating if the abscissa range is to be set according to the framebreak range because the two variables are identical

FRNGE1 lower limit of current framebreak range

FRNGE2 upper limit of current framebreak range

SRNGE1 lower limit of current selection range

SRNGE2 upper limit of current selection range

ALLPNT logical flag indicating if no selection testing is required

NOSET logical flag indicating if the framebreak range is to be set by the program

NOFRNG logical flag indicating if the framebreak range has not yet been set by the program

NOSEL logical flag indicating if the selection range is to be set by the program

NOSRNG logical flag indicating if the selection range has not yet been set by the program

RANGUP increment between frames for the abscissa variable when SETRNG = .TRUE.

XDCNT counter for abscissa variable

YDCNT counter for ordinate variable

K counter for output buffers XX and YY

IRECS number of current record

NEEDID logical flag indicating if frame identification title is needed

IFRME frame counter

ICOUNT number of segments in this record

IGO integer flag indicating action to be taken after current frame ends (1 = terminate job; 2 = continue)

IREC$ number of requested record. Reset by GETREC if an end-of-file or error occurs

FRSTFF logical flag indicating if this is the first check of the framebreak variable for this record

FRSTSF logical flag indicating if this is the first check of the selection variable for this record

TEMP8 variable used for temporary storage of value returned by VALUGP

IXY used to determine if this point meets the requirements of XFREQ and YFREQ(I)

FRSTFV logical flag indicating if this is the first check of the framebreak variable for this variable segment

FIRSTX logical flag indicating if the X-variable in a fixed segment has been determined

BRKVAL current value of the framebreak variable

JGO integer flag indicating reason for dumping current buffers

1 = x-value out of specified range

2 = y-value out of specified range

3 = buffer full

4 = framebreak value out of range

5 = end of job

N1 number of first ordinate buffer to be dumped

N2 number of last ordinate buffer to be dumped

IND index of either framebreak or selection variable

FBREAK check value of framebreak variable, used when NOSET = .TRUE.

SELTMP check value of selection variable, used when NOSEL = .TRUE.

INDEX location of frame identification variable

FRMVAR value of frame identification variable

NN length of EBCDIC label for current value of frame variable

IRECF number of first record used for this frame

DEVS array of logical flags indicating plotting devices to be used for this frame

IRECB number of bad records read for this frame

ILINE number of lines used on this page for read error messages

IPAGE number of current summary page

XOLD previous value of x-variable

XCYCLE logical flag indicating if cyclic x-values are expected

IWORD check value of the current framebreak variable

JWORD current value of the current framebreak variable

IPRMOD integer array of printer plot requirements set by CONCRD

INTREL logical flag indicating if the y-values used for interpolation are relative values

COEF array of coefficients for the Lagrange interpolation polynomial for this record

YVAL transformed value of the independent variable on which interpolation is to be performed

YA $(YVAL)^{i-1}$, used to evaluate the Lagrange interpolation polynomial

XA transformed value of the dependent variable, determined by interpolation

X untransformed value of the dependent variable

IFMREP number of current summary printout

C. Equivalences

(JWORD, JBYTE(1))	used for byte-by-byte comparisons
(IWORD, IBYTE(1))	used for byte-by-byte comparisons
(INTY(1), INTY1, FINTY1)	used to obtain location of interpolation variable
(INTY(2), INTY2)	used to obtain length of interpolation variable
(INTY(3), INTY3)	used to obtain type of interpolation variable
(XD(1), IX1)	used to obtain location of x-variable
(XD(2), IX2)	used to obtain length of x-variable
(XD(3), IX3)	used to obtain type of x-variable
(RFMT, FRMT(1))	used to set up format for frame identification values

VII. Generated messages:

A. Summary printout for each frame.

B. Read error on input data set:

MODULE GPFPS I/O ERROR ON RECORD iiii PAGE jjjj

(error message from FTIO or DAIO, as appropriate)

C. No points found for this plot, so no frame output although a summary listing was generated.

NO PLOT FOR THIS FRAME

D. End of file on input data set:

***** END OF USABLE INPUT DATA *****

E. End of job

END OF GPFPS RUN PAGE jjjj

VIII. Constraints: none

4.2 CONCRD

I. Abstract: subroutine CONCRD reads the input control data set, and initializes the program execution parameters as specified by the input keywords.

II. Background:

A. Author: G. Masaki, G. Mason, C. McQuillan

B. Date completed: 7 May 1976

C. Source language: FORTRAN H

D. Program size: 10376 bytes

III. Requirements:

A. Calling sequence

CALL CONCRD (ICODE, IPRMOD)

where

ICODE indicates type of return from this subroutine (output)

0 = normal return

-1 = error in control data set forcing termination of job.

IPRMOD is an array of printer plot requirements (output)

IPRMOD(1) = printer plot output unit from MODE(15)

IPRMOD(2) = printer plot frame size from MODE(14)

B. Calling routines:

GPFPS

C. COMMON blocks:

INPUT$, PLOTGP$, IO

D. Non-system routines:

CRNCH1, CRNCH3, VALUGP, DGITGP, GETID, CWDGP

The following are contained in the Wolf Plot and Contour Package:

PRINTR, NOW, DATE

IV. Method: see section V.

V. Program flow:

A. Initialize variables and obtain the run ID for this job. List default values for all keywords.

B. Read a control card, and list it on the printer.

1. If it is a comment card, go back to setp B.

2. Otherwise, remove unnecessary blanks and place on EBCDIC '$' at end of string.

3. If an end-of-file is encountered, go to step E.

C. Process all keywords on this control card.

1. Determine type and length of keyword.
2. If the keyword is illegal, print a message and go to step 7.
3. Process keywords of form WORD = value, where WORD may be subscripted. Go to step C7.
4. Process keywords of form WORD = A, B, C, where WORD may be subscripted. Go to step C7.
5. Process keywords of form WORD = 'string', where WORD may be subscripted. Go to step C7.
6. Process mode settings for plotter specifications.
7. Go to step B if this is the last keyword. Otherwise, go to step C1.

D. If an error occurs in a keyword specification, write a message and go to step B.

E. When all control cards have been processed, list all keywords as they will be used during program execution. Return to calling program.

VI. Glossary of labels:

A. Constants

TYPE4$(I) logical flag indicating whether the Ith keyword may be indexed

LENGTH(I) length of an element for the Ith keyword

POINTC(I) length of an element for the Ith keyword for hollerith values

POINTR(I) starting byte in COMMON block INPUT$ for the Ith keyword

PLOTPS plotter numbers according to WPCP conventions

PLOTDG plotter digit value according to WPCP conventions

The following are EBCDIC characters:

ASTRK '*'

BLANK ' '

COMMA ','

COLON ':'

DOLLAR '$'

EQUAL '='

F 'F'

LFTPAR '('

ONE '1'

QUOTE 1H'

RGTPAR ')'

SEMCOL ';'

T 'T'

ZERO 'Ø'

B. Variables

DEFALT logical flag indicating type of keyword values being listed.

.TRUE. = print default values before processing control data set

.FALSE. = print values specified on input, and default values for those keywords not specified, as they will be used during program execution

INPUT	array of characters on control card current being processed
INDEX1, INDEX2	indices used in deleting blanks from INPUT
HOLRTH	logical flag indicating whether blanks are to be deleted from INPUT
IPOINT	pointer to character in INPUT being processed
IPOS	position pointer returned by CRNCH1, CRNCH3, CWDGP, or VALUGP.
TYPE4	logical flag indicating whether the keyword currently being processed may be indexed
ITYPE	number of current keyword in keyword table, as returned by subroutine CWDGP
INDEX	index number for indexed keywords. Set to 1 for non-indexed keywords
INDEX1	position in COMMON block INPUT$ at which the processed values for the current keyword are to be placed

ERR	logical flag indicating whether errors were found in the input control data set
DDATE	EBCDIC string containing current month, day, and year
IYDD	current date (YYDDD) in integer form
IHR	hour of the day
IMIN	minute of the hour
IHM	hundredths of seconds of the minute
ICLASS	integer flag indicating type of values associated with the current keyword 1 = positive integers 2 = positive or negative integers 3 = real values 4 = integer or real values
INCRE	maximum permissible length in characters of current keyword
INDEX	pointer for array position in COMMON block INPUT$
IPS	pointer to digit being evaluated or modified.
IND	index for dashed, multiple, or overplotted lines

C. Equivalences

(RECFM, FARRAY(1), IARRAY(1), CHAR(1)) used to obtain REAL*4, INTEGER*4, and LOGICAL*1 values of COMMON block INPUT$

(NOCHAR(1), NSYMB(1)) used to equate all symbol counters

(TITLEY(1, 3), TITLY(1, 1)) used for printout

(FORM(1, 1), FRMBRK(1, 2)) used to determine formats

(INPUT(1), INPUT8(1)) used to obtain REAL*8 and LOGICAL*1 values of character string on the current control card.

VII. Generated messages:

A. Table of default values for control keywords.

B. Table of execution-time values for control keywords.

C. Listing of control data set in card-image format.

D. Control data set error message:

**MODULE CONCRD...ERROR IN CONTROL CARD...SCAN POINTER = approximate location of error. VARIABLE IS TYPE number in Table I.

E. Execution termination message:

MODULE CONCRD EXECUTION TERMINATING DUE TO ERRORS IN CONTROL CARD INPUT

VIII. Constraints: none

4.3 CWDGP

I. Abstract: subroutine CWDGP determines the length of a given keyword and its position in a table of permissible keywords.

The position is the number of the keyword given in Table I.

II. Background:

A. Author: G. Masaki, G. Mason

B. Date completed: 14 December 1971

C. Source language: FORTRAN H

D. Program size: 716 bytes

III. Requirements:

A. Calling sequence:

CALL CWDGP (CHAR, IPOS, ITYPE)

where

CHAR = keyword in a LOGICAL*1 array (input)

IPOS = length of keyword (output)

ITYPE = table location of keyword (output)

(ITYPE = 0 if keyword not in table)

B. Calling routines:

CONCRD

C. COMMON blocks: none

D. Non-system routines: none

IV. Method: the keyword is compared to values in a table of permitted keywords.

V. Program flow: see section IV

VI. Glossary of labels:

A. Constants

CONTRL table of keywords in EBCDIC form

BLANK

B. Variables: none

C. Equivalences

(CONTROL(1), CONBYT(1, 1) used to obtain keywords in LOGICAL*1 form for comparison with CHAR

VII. Generated messages: none

VIII. Constraints: none

4.4 GETREC

I. Abstract: subroutine GETREC gets the next data record from the input device using the proper access method.

II. Background:

A. Author: G. Mason

B. Date completed: 26 June 1972

C. Source language: FORTRAN H

D. Program size: 458 bytes

III. Requirements:

A. Calling sequence

CALL GETREC (IREC)

where

IREC = number of the record to be read in (input)

On return IREC = 0 if an end-of-file has been read

IREC = -IREC if a read error has occurred.

B. Calling routines:

GPFPS, LEGRGP

C. COMMON blocks:

INPUT$, READ$$, IO

D. Non-system routines:

FREAD from FTIO

DREAD from DAIO

IV. Method: see section V.

V. Program flow:

A. If the requested record is already in core, return.

B. For sequential organization, read in records one at a time until the requested record is in core.

1. If an end-of-file is encountered, set IREC = 0 and return.

2. If a read error occurs before the requested record is read, continue reading.

3. If a read error occurs on the requested record, set IREC = -IREC, and return.

C. For direct access organization, read the requested record into core. If a read error occurs, set IREC = -IREC and return.

VI. Glossary of labels:

A. Constants: none

B. Variables

IREC number of the requested record

IREC1 number of the record currently in core

C. Equivalences: none

VII. Generated messages: none

VIII. Constraints

For data sets with sequential organization, records must be requested in ascending order, as the input data set is not rewound during program execution.

4.5 GRIDGP

I. Abstract: subroutine GRIDGP has three entry points. GRIDGP initializes the plot frame and outputs any titles and grid overlays. PLOTGP plots the selected data on the appropriate grid. ENDGP outputs summary labels and terminates the frame.

II. Background:

A. Author: G. Masaki, G. Mason, C. McQuillan

B. Date completed: 7 May 1976

C. Source language: FORTRAN H

D. Program size: 13196 bytes

III. Requirements:

A. Calling sequences:

1. Call GRIDGP (DEVS)

where DEVS = a LOGICAL*1 array of plot selector switches (input)

When DEVS (I) = .TRUE., the corresponding plotter is used, as follows:

1 printer

2 SC4020

3 Gerber

4 Calcomp

5 SD4060

2. CALL PLOTGP (X, Y, N, J, FIRST)

where

X, Y = singly-subscripted arrays containing the coordinates of the points to be plotted (input).

N = number of points to be plotted (input).

J = index number of the Y variable being plotted (input)

$1 \leq J \leq 10$

FIRST = logical switch indicating if this is the first part of a string of X, Y arrays (input).

3. CALL ENDGP

B. Calling routines:

GPFPS

C. COMMON blocks:

INPUT$, PLTGP$

D. Non-system routines: The following are all contained in the Wolf Plot and Contour Package:

PLOTST, IDFRME, DIAGLN, VECTOR, PLOTS, AFFINE, GRID, PLOT, EDIT

IV. Method: see section V.

V. Program flow:

GRIDGP

A. If this is the first call to GRIDGP, initialize the WPCP, output an ID frame, initialize internal variables.

B. Output various titles on required plotters.

C. If there is only one Y variable, or the grids are to be stacked, output the grid(s).

D. Otherwise, there are to be multiple Y-variables on the same grid.

If all Y scales are the same, output a single grid

Otherwise, output a single grid with multiple Y-axes to the left of it.

E. Return to calling program.

PLOTGP

A. Set up scaling factors through calls to routines in WPCP.

B. Plot the data for one Y-variable, using points, single lines, or multiple lines as requested.

C. Return to calling program.

ENDGP

A. Output the following information in the upper section of the plot: frame number, time, off-scale points, numbers of points.

B. Terminate the frame by a call to the WPCP.

C. Return to calling program.

VI. Glossary of labels:

A. Constants

COMMA

BLANK

The following determine the size of grid overlays:

F50	bottom of grid
F1000	right edge of grid
F900	top of grid
F475	midpoint of y-axis
F525	midpoint of x-axis
YTOP$	top of first stacked grid
YDIF$	increment used between stacked grids
I25	increment for each y-axis

B. Variables:

INIT	frame counter
NXL	number of intervals between labeled lines on x-axis

NYL	number of intervals between labeled lines on y-axis
FMTX	abscissa labeling format
FMTY	ordinate labeling format
XOO	left edge of grid for multiple grids
XO	x location of current y-axis
S1	left edge of x-axis in object space
S2	lower edge of y-axis in object space
S3	right edge of x-axis in object space
S4	upper edge of y-axis in object space
YTOP	top of current grid, for stacked grids
YDIF	total size of grid including spacing, for stacked grids
YBOT	bottom of current grid, for stacked grids.
K	number of overplots in data being plotted
TIME	generated title output above plot
FRAMES	generated title output above plot
PNTOFF	generated title output above plot.

C. Equivalences

(FORM(1, 1), FRMBRK(1, 2)) used to determine formats

VII. Generated messages: none

VIII. Constraints: none

4.6 LEGRGP

I. Abstract: subroutine LEGRGP determines the coefficients of the Lagrangian interpolation polynomial of requested order for a given record. Entry point SETINT initializes the subroutine, and sets up a temporary data set on intermediate storage to be used in determining the interpolation coefficients.

II. Background:

A. Author: G. Masaki, G. Mason

B. Date completed: 21 April 1972

C. Source language: FORTRAN H

D. Program size: 6592 bytes

III. Requirements:

A. Calling sequence

1. CALL LEGRGP (POLY, IREC, FX, XSCALE, XSHIFT, YSCALE, YSHIFT)

where:

POLY = array containing the coefficients (low order to high) of the interpolation polynomial (output).

IREC = number of record for which coefficients are to be determined (input).

IREC = 0 if an error occurs (output)

FX = value of X variable at start of current record (output).

XSCALE = scaling factor for X values (output).

XSHIFT = shift factor for X values (output).

YSCALE = scaling factor for Y values (output).

YSHIFT = shift factor for Y values (output).

2. CALL SETINT

B. Calling routines:

GPFPS

C. COMMON blocks:

INPUT$, READ$$, IO

D. Non-system routines:

DREAD from DAIO

FREAD, REWIND from FTIO

CDEFGP, VALUGP

IV. Method: Lagrangian interpolation procedure

Given: a function $y = f(x)$ defined at n+1 points (X_0, Y_0), $(X_1, Y_1), \ldots (X_n, Y_n)$

Define: $L(x)$ = nth degree polynomial through these points

Determine the coefficients at L:

$$L(x) = \frac{(x-x_1)(x-x_2)\ldots(x-x_n)}{(x_0-x_1)(x_0-x_2)\ldots(x_0-x_n)} Y_0 + \ldots + \frac{(x-x_0)(x-x_1)\ldots(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)\ldots(x_n-x_{n-1})} Y_n$$

$$= \left(\sum_{i=1}^{n+1} k_{0i}\, x^{i-1}\right) \frac{Y_0}{(x_0-x_1)(x_0-x_2)\ldots(x_0-x_n)} + \ldots +$$

$$\left(\sum_{i=1}^{n+1} k_{ni}\, x^{i-1}\right) \frac{Y_n}{(x_n-x_1)(x_n-x_2)\ldots(x_n-x_{n-1})}$$

Then

$$L(x) = \sum_{i=1}^{n+1} C_i\, x^{i-1}$$

where

$$C_1 = k_{01} \frac{Y_0}{(x_0-x_1)(x_0-x_2)\cdots(x_0-x_n)} + \cdots + k_{n1} \frac{Y_n}{(x_n-x_1)(x_n-x_2)\cdots(x_n-x_{n-1})}$$

$$C_{n+1} = k_{0,n+1} \frac{Y_0}{(x_0-x_1)(x_0-x_2)\cdots(x_0-x_n)} + \cdots + k_{n,n+1} \frac{Y_n}{(x_n-x_1)(x_n-x_2)\cdots(x_n-x_{n-1})}$$

The coefficients k_{ij} are determined by the subroutine COEFGP. Subroutine LEGRGP determines the interpolation coefficients C_i, and stores them in the array POLY.

For ease of calculation the values x_i and y_i are transformed to the range -1 to 1. If primes denote the original data values, then

$$x = (x' - \text{xshift}) \cdot \text{xscale} \qquad y = (y' - \text{yshift}) \cdot \text{yscale}$$

$$\text{xshift} = \tfrac{1}{2}(x'_{max} + x'_{min}) \qquad \text{yshift} = \tfrac{1}{2}(y'_{max} + y'_{min})$$

$$\text{xscale} = 2/(x'_{max} - x'_{min}) \qquad \text{yscale} = 2/(y'_{max} - y'_{min})$$

Interpolation coefficients are determined from the pairs of transformed points (x,y). The values for which interpolation is performed must be transformed in a similar manner.

Thus, given the particular value x'_p of the independent variable,

$$x_p = (x_p' - \text{xshift}) \cdot \text{xscale}$$

The corresponding value y_p of the dependent variable is given by

$$y_p = L(x_p)$$

$$= \sum_{i=1}^{n+1} C_i \, x_p^{i-1}$$

where $y_p' = y_p/yscale + yshift.$

V. Program flow:

LEGRGP

A. Select the set of (x, y) pairs to be used to determine the coefficients of the interpolation polynomial for the current record. (see section 2.3, items 27, 28, 29)

B. Read these values from intermediate storage on unit 19, and determine XMIN, XMAX, XSCALE, XSHIFT and YMIN, YMAX, YSCALE, YSHIFT.

C. Transform the (x, y) pairs.

D. Determine the coefficients k_{ij}, which are stored in array C.

E. Form the coefficients C_i, which are stored in array POLY.

F. Return to calling program, where interpolation is performed for required values of the independent variable, and the resulting value of the dependent variable is transformed back to the original range.

G. A read error will terminate program execution.

SETINT

A. Read the input data set from first to last requested record, using the proper access method.

B. Store the pairs of (x, y) values in two buffers of 2000 bytes each.

C. Output the buffers onto intermediate storage on unit 19. The final buffer may be only partially filled.

D. See various counters and pointers, and return.

E. If a read error occurs, a message is printed, and processing continues.

VI. Glossary of labels:

A. Constants

MMAX maximum number of interpolation values allowed in the intermediate buffer

NFPOS relative number of first record in a physical block from intermediate storage

B. Variables

NL number of the record in core

L number of records following current record to be used for interpolation

NREC number of physical blocks written in intermediate storage for interpolation data set

MREC number of the physical block from intermediate storage currently in core

N number of the record from the current physical block currently in buffer

XVAL buffer of x-values from the current physical block

YVAL buffer of y-values from the current physical block

X array of x-values to be used for current interpolation

Y — array of y-values to be used for current interpolation

M — number of records in last physical block

NF — number of first record from current physical block

IND — number of first record from current physical block to be used for the current interpolation

YMIN — minimum value of array X

YMAX — maximum value of array X

YMIN — minimum value of array Y

YMAX — maximum value of array Y

FACTOR — $\prod_{\substack{j=1 \\ j\neq i}}^{order1} Y(I)/(X(I)-X(J))$ used in determining the coefficients k_{ij}

XX — array of x-values ($I \neq J$) used in determining the coefficients k_{ij}

C — array of coefficients k_{ij}

ORDER1 — number of interpolation coefficients C_i to be determined

IA — number of first record from input data set used to form interpolation data set

IB number of last record from input data set used to form interpolation data set

IC number of first record required from input data set for interpolation

C. Equivalences

(XD(1), XD1), (XD(2), XD2), (XD(3), XD3) used to obtain Y-values for interpolation

(INTX(1), INTX1), (INTX(2), INTX2),(INTX(3), INTX3) used to obtain x-values for interpolation

(XV(11), XVAL(1)) used in buffering the x-values

(YV(11), YVAL(11) used in buffering the y-values

VII. Generated messages:

LEGRGP

A. Read error on interpolation data set

MODULE LEGRGPREAD ERROR ON INTERPOLATION COEFFICIENT DATA SET. JOB WILL TERMINATE.

SETINT

B. Read error in forming interpolation data set

MODULE LEGRGPREAD ERROR ON RECORD i OF PLOT DATA SET. INTERPOLATION DATA WILL BE FETCHED FROM BUFFER AS READ

VIII. Constraints

SETINT must be called before the first call to LEGRGP. There may not be INTORD+1 consecutive identical X or Y values.

4.7 COEFGP

I. Abstract: subroutine COEFGP determines the coefficients of the nth order polynomial represented by the product $\prod_{i=1}^{n} (x-x_i)$. The x_i are given.

II. Background:

A. Author: G. Masaki

B. Date completed: 27 August 1971

C. Source language: FORTRAN H

D. Program size: 452 bytes

III. Requirements:

A. Calling sequence

```
CALL COEFGP (C, X, N)
```

where

C = array of coefficients, low order first (output)

X = array of values (input)

N = number of elements in X (input)

$$\prod_{i=1}^{N} (Y-X_i) = \sum_{i=1}^{N+1} C_i Y^{i-1}$$

B. Calling routines

LEGRGP

C. COMMON blocks: none

D. Non-system routines: none

IV. Method: n successive multiplications are performed. The jth multiplication involves the jth root, and generates j+1 coefficients. Denote the contribution to the ith coefficient

by all roots up to the jth by $k_{i,j}$. Then

$$k_{i,j} = k_{i-1,\,j-1} - k_{i,j-1}\,x_j \qquad \begin{matrix} j = 1, 2, \ldots, n \\ i = 1, 2, \ldots, j, j+1 \end{matrix}$$

where: $k_{1,0} = \begin{cases} 0, & 1 \neq 1 \\ 1, & 1 = 1 \end{cases}$

$$k_{0,1} = 0$$

$$k_{j+1,j-1} = 0$$

The ith coefficient of the product is therefore $k_{i,n}$.

EXAMPLE $(X-X_1)(X-X_2)(X-X_3) = C(4)\,X^3 + C(3)\,X^2 + C(2)\,X + C(1)$

$n = 3$

j=1 i=1 $k_{1,1} = k_{0,0} - k_{1,0}\,x_1 = -x_1$

i=2 $k_{2,1} = k_{1,0} - k_{2,0}\,x_1 = 1$

j=2 i=1 $k_{1,2} = k_{0,1} - k_{1,1}\,x_2 = x_1x_2$

i=2 $k_{2,2} = k_{1,1} - k_{2,1}\,x_2 = -(x_1 + x_2)$

i=3 $k_{3,2} = k_{2,1} - k_{3,1}\,x_2 = 1$

j-3 i=1 $k_{1,3} = k_{0,2} - k_{1,2}\,x_3 = -x_1x_2x_3 = C(1)$

i=2 $k_{2,3} = k_{1,2} - k_{2,2}\,x_3 = x_1x_2 + x_3(x_1 + x_2) = C(2)$

i=3 $k_{3,3} = k_{2,2} - k_{3,2}\,x_3 = -(x_1 + x_2 + x_3) = C(3)$

i=4 $k_{4,3} = k_{3,2} - k_{4,2}\,x_3 = 1 = C(4)$

V. Program flow: see section IV.

VI. Glossary of labels:

A. Constants: none

B. Variables:

XX Temporary save for X(I)

CSAVE1 Temporary save for C(J)

CSAVE2 Temporary save for CSAVE1

C. Equivalences: none

VII. Generated messages: none

VIII. Constraints:

N < 6

4.8 VALUGP

I. Abstract: subroutine VALUGP determines the value represented by a string of bytes, either binary or EBCDIC, for real or integer values.

II. Background:

A. Author: G. Masaki

B. Date completed: 31 August 1971

C. Source language: FORTRAN H

D. Program size: 638 bytes

III. Requirements:

A. Calling sequence

CALL VALUGP (CHAR, N, ITYPE, RESULT)

where

CHAR = string to be interpreted in a LOGICAL*1 array (input)

N = number of bytes in CHAR to be used (input)

ITYPE = type of variable represented by CHAR (input)
coded as follows:
1 = integer of length 2 or 4 bytes
2 = real of length 4 or 8 bytes
3 = EBCDIC in I, F, E, or D format

RESULT = resulting value in REAL*8 form (output)
If errors are encountered, RESULT = 'ERROR'

B. Calling routines

GPFPS, CONCRD, LEGRGP

C. COMMON blocks: none

D. Non-system routines: CRNCH1

V. Method: The result is obtained through equivalences, for ITYPE = 1 or 2, or by calling CRNCH1, for ITYPE = 3.

VI. Program flow:

A. CHAR is stored in a buffer which is equivalenced with REAL*8, REAL*4, INTEGER*4, and INTEGER*2 words.

B. If ITYPE = 1 or 2, the proper value is placed in RESULT according to N and ITYPE.

C. If ITYPE = 3, CRNCH1 is used to decode CHAR into RESULT. If an error return is executed from CRNCH1, RESULT is set to the character string 'ERROR'.

D. Return to calling program.

VII. Glossary of labels:

A. Constants: none

B. Variables:

BUFFER LOGICAL*1 buffer to contain CHAR

BYTES same as BUFFER

INT$4 INTEGER*4 array

INT$2 INTEGER*2 array

REAL$4 REAL*4 array

REAL$8 REAL*8 array

C. Equivalences

(BUFFER(1), BYTES(1), REAL$8(1), REAL$4(1), INT$4(1), INT$2(1)) used to obtain result in desired form, for ITYPE = 1 or 2.

VII. Generated messages: none

VIII. Constraints

$N \leq 20$

4.9 DGITGP

I. Abstract: subroutine DGITGP sets a specified digit of a number to a given value.

II. Background:

A. Author: G. Masaki

B. Date completed: 19 September 1971

C. Source language: FORTRAN H

D. Program size: 364 bytes

III. Requirements:

A. Calling sequence

CALL DGITGP (IVALUE, IPOS, IDIG)

where

IVALUE = value to be changed (input); changed value (output)

IPOS = decimal digit to be set, numbered from right to left (input)

IDIG = value to be set into IPOS digit of IVALUE

B. Calling routines

CONCRD

C. COMMON blocks: none

D. Non-system routines: none

IV. Method: the value of the number IVALUE with digit IPOS set to zero is determined, and IDIG is inserted in its place.

In integer arithmetic:

$$IVALUE = (IVALUE/10^{IPOS}) - 10^{IPOS} + IVALUE - (IVALUE/10^{IPOS-1})*10^{IPOS-1} + IDIG*10^{IPOS-1}$$

V. Program flow: see section IV.

VI. Glossary of labels:

A. Constants: none

B. Variables:

NUMB = 10^{IPOS-1}

NUMB1 = 10^{IPOS}

C. Equivalences: none

VII. Generated messages: none

VIII. Constraints:

IPOS > 0

$9 \geq IDIG \geq 0$

IVALUE > 0

4.10 CRNCH1

I. Abstract: subroutine CRNCH1 determines, from a string of EBCDIC characters representing a number in an I, E, D, or F format, the value represented, and the number of characters in the string. No imbedded blanks are allowed in the string.

II. Background:

A. Author: G. Masaki

B. Date completed: 31 August 1971

C. Source language: FORTRAN H

D. Program size: 1220 bytes

III. Requirements:

A. Calling sequence

CALL CRNCHI (CHAR, VALUE, IPOS, ¢#)

where

CHAR = string of EBCDIC characters containing value to be determined (input)

VALUE = REAL*8 value (output)

IPOS = number of characters in input string (output)

= error return label number

B. Calling routines

VALUGP, CRNCH3, CONCRD

C. COMMON blocks: none

D. Non-system routines: none

IV. Method: see section V.

V. Program flow:

A. Determine sign factor (±1).

B. Evaluate the left part (integer) of number.

C. Evaluate the right part (fraction) of number.

D. If exponential notation (D or E format) is used, compute and apply the exponent.

E. Multiply in the sign factor.

F. The error return is executed whenever a byte which is assumed to contain an EBCDIC digit is not in the decimal range 240 to 249 (hex FØ to F9). IPOS is set to 1.

G. Return to calling program.

VI. Glossary of labels:

A. Constants: the following are LOGICAL*1 characters (EBCDIC)

POINT	'.'
E	'E'
D	'D'
MINUS	'-'
PLUS	'+'

B. Variables

NEGTIV	sign factor for VALUE (±1)
FACTOR	multiplicative factor for decimal digits
INFACTR	sign factor for exponent
IEXP	exponent value

C. Equivalences

(IWORD, BYTES(1)) used to convert numbers from EBDIC to binary

VII. Generated messages: none

VIII. Constraints

A. No imbedded blanks are allowed in the input string.

B. For integer value greater than 10^4, VALUE should be converted from REAL*8 to INTEGER*4 in the calling program.

4.11 CRNCH3

I. Abstract: subroutine CRNCH3 extracts values from an EBCDIC string of characters in the form 'XXX, YYY, ZZZ'. XXX, YYY, ZZZ represent numbers in I, E, D, or F format, and must contain no imbedded blanks. YYY may be input as the character string 'LAST', in which case the value returned is 10^9.

II. Background:

A. Author: G. Masaki

B. Date completed: 31 August 1971

C. Source language: FORTRAN H

E. Program size: 546 bytes

III. Requirements:

A. Calling sequence

CALL CRNCH3 (CHAR, VALUE, IPOS, ¢#)

where

CHAR = string of EBCDIC characters as described in abstract (input)

VALUE = array of 3 REAL*8 numbers returned as binary values (output).

IPOS = length of CHAR (output).

= error return label number.

B. Calling routines

CONCRD

C. COMMON blocks: none

D. Non-system routines: CRNCH1

IV. Method: see section V.

V. Program flow:

A. Interpret first item in string through use of CRNCH1.

B. If no comma follows first item, return.

C. Inspect first character of record item.

If 'L', which means that the second item is 'LAST', set value to 10^9.

Otherwise, call CRNCH1.

D. If no comma follows second item, return.

E. Interpret third item through use of CRNCH1, and return.

F. An error return from CRNCH1 results in the execution of the error return from CRNCH3.

VI. Glossary of labels:

A. Constants: the following are LOGICAL*1 characters (EBCDIC)

L	'L'
COMMA	','

B. Variables

INDEX	position counter

C. Equivalences: none

VII. Generated messages: none

VIII. Constraints: the input string may not contain imbedded blanks.

4.12 COMMON blocks

A. INPUT$: 2048 bytes

1. Specification

```
COMMON/INPUT$/
     · RECFM, RECSIZ, FIXSIZ, VARSIZ, COUNT(3),
     · XD(3), YD(3, 10), DATLOC, XFREQ(3), YFREQ(3, 10),
     · IPLOT(3, 8), AFF(4), LOGNEG(4), SIZLET, DASH(4, 10),
     · NODASH(10), MDELTA(4, 10), NOMULT(10), LINTYP(10), MLTPLE(10),
     · DEVICE, IOUT6, FRMBRK(3, 10), FRMRNG(3), RANGEX(3),
     · RANGEY(3, 10), LINE(10), STACK, SYMBOL(12), TITLEX(40),
     · TITLEY(40, 10), MODEX, MODEY(10), RECUSE(3), MAXFRM,
     · SELECT(3), SELRNG(3), INTX(3), INTY(3), INTORD,
     · RUNID(8), JOBTTL(40), FRMEID(3, 10), NVAR, NSYMB(10),
     · NTTLX, NTTLY(10), NJOBTL, NFRMID, INTERP,
     · DSORG
     LOGICAL*1 FORM(8, 11)
     EQUIVALENCE (FRMBRK(1, 2), FORM(1, 1))
```

2. Description of variables

Name	Type	Description
RECFM	INTEGER*4	*
RECSIZ	INTEGER*4	*
FIXSIZ	INTEGER*4	*
VARSIZ	INTEGER*4	*
COUNT	INTEGER*4	*
XD	INTEGER*4	*
YD	INTEGER*4	*

Name	Type	Description
DATALOC	INTEGER*4	*
XFREQ	INTEGER*4	*
YFREQ	INTEGER*4	*
IPLOT	INTEGER*4	information for plotting devices IPLOT(I, J) I indicates device: 1 = printer 2 = SC4020; 3= Gerber; 4 = Calcomp; 5 = SD4060; 6, 7, 8 = not used. J indicates frame output: 1 = start frame; 2 = end frame; 3 = increment between frames.
AFF	REAL*4	arguments to subroutine AFFINE (section 3.9 of WPCP)
LOGNEG	LOGICAL*1	arguments to subroutine NEGLOG (section 4.2 of WPCP)
SIZLET	REAL*4	argument to subroutine SETSIZ (section 5.13 of WPCP)
DASH	REAL*4	argument to subroutine PLOTS (section 8 of WPCP)
NODASH	INTEGER*4	same as above
MDELTA	REAL*4	same as above
NOMULT	INTEGER*4	same as above
LINTYP	LOGICAL*4	flag indicating whether line types are to be used for the corresponding ordinate

Name	Type	Description
MLTPLE	INTEGER*4	number of overplots for the corresponding ordinate
DEVICE	INTEGER*4	argument to subroutine PLOTST (section 1.1 of WPCP)
IOUT6	INTEGER*4	argument to subroutine PRUNIT (section 11.4 of WPCP)
FRMBRK	INTEGER*4	*
FORM	LOGICAL*1	format for labeling X-axis (FORM(J, 1), J = 1,8) format for labeling Ith Y-axis (FORM(J, I+1), J = 1,8)
FRMRNG	REAL*4	*
RANGEX	REAL*4	*
RANGEY	REAL*4	*
LINE	INTEGER*4	*
STACK	INTEGER*4	*
SYMBOL	LOGICAL*1	*
TITLEX	LOGICAL*1	*
TITLEY	LOGICAL*1	*
MODEX	INTEGER*4	*
MODEY	INTEGER*4	*
RECUSE	INTEGER*4	*
MAXFRM	INTEGER*4	*
SELECT	INTEGER*4	*
SELRNG	REAL*4	*

Name	Type	Description
INTX	INTEGER*4	*
INTY	INTEGER*4	*
INTORD	INTEGER*4	*
RUNID	LOGICAL*1	*
JOBTTL	LOGICAL*1	*
FRMEID	INTEGER*4	*
NVAR	INTEGER*4	number of ordinates
NSYMB	INTEGER*4	number of characters in SYMBOL
NTTLX	INTEGER*4	number of characters in TITLEX
NTTLY	INTEGER*4	number of characters in TITLEY
NJOBTL	INTEGER*4	number of characters in JOBTTL
NFRMID	INTEGER*4	number of frame ID's
INTERP	LOGICAL*4	flag indicating whether interpolation is to be performed
DSORG	INTEGER*4	*

* Contains value of corresponding control keyword

** (INTY(1) will be REAL*4 if INTY(3) = 4

B. READ$$: 32000 bytes

1. Specification

   COMMON/READ$$/BUFFER (32000)

2. Description of variables

Name	Type	Description
BUFFER	LOGICAL*1	array containing one physical record from input data set on unit 20

C. PLTGP$: 252 bytes

1. Specification

COMMON/PLTGP$/POINTS(10), IFRME, OFF(6), IDPNT, FRMTTL(160), DDATE(8), IHR, IMIN, ISEC

2. Description of variables

Name	Type	Description
POINTS	INTEGER*4	number of points for each ordinate variable
IFRME	INTEGER*4	frame number
OFF	INTEGER*4	number of rejected points in Ith category 1: < RANGEX 2: > RANGEX 3: < RANGEY 4: > RANGEY 5: < SELRNG 6: > SELRNG
IDPNT	INTEGER*4	number of characters in FRMTTL
FRMTTL	LOGICAL*1	frame ID title
DDATE	LOGICAL*1	date in form MM/DD/YY
IHR	INTEGER*4	current hour of day
IMIN	INTEGER*4	current minute of hour
ISEC	INTEGER*4	current second of minute

D. IO: 16 bytes

1. Specification

COMMON/IO/IN, IP, IPR, INCD

2. Description of variables

Name	Type	Description
IN	INTEGER*4	FORTRAN unit 20, on which input data set is to be read
IP	INTEGER*4	FORTRAN unit 19, on which data is stored temporarily when interpolation is requested
IPR	INTEGER*4	FORTRAN unit number on which printer plots are to be output
INCD	INTEGER*4	FORTRAN unit 5, on which control data set is input

4.13 Assembler routines

The General Purpose Film Plotting System includes five supporting subroutines written in OS Assembler Language. All have been assembled under the Level G version of that language, and are described below:

CONMOV sets the specified number of elements in an array to a specified constant value. It is used by the main program to zero out two arrays.

FRONT, called by the main program, outputs on the printer two pages of job identification information which marks the start of the execution summary log. The program name, date and time of execution, and programmer identification are all obtained from the operating system by assembler programs indicated below:

BLKLET formats the information on these header pages in block letters. Each letter is 13 printer spaces wide, with 5 spaces between each letter. A maximum of 7 letters are printed per line.

DAYCNT determines the month and day of month from the day of year, which is provided by the operating system.

GETID obtains the jobname from the operating system. The first 5 letters are the programmer identification.

REQUEST FOR APPROVAL FOR PUBLICATION OF THE MANUSCRIPT

(See Reverse for Detailed Instructions) (Refer to GMI 2220.5, Approval and Special Requirements for GSFC Publications)

1. X-Document Number	2. Title
X-626-77-163	General Purpose Film Plotting System
ALL COSTS TO BE CHARGED TO	
3. GSFC job order number 620-992-66-01-01	Typed name and signature of Author(s) Christine McQuillan Christine McQuillan
NASA agency-wide code designation	

4 APPROVAL REQUESTED AS

- ☐ Presentation
- ☐ Journal Publication
- ☑ Formal NASA Report
- ☐ CR Publication
- ☐ Working Paper
- ☒ Preprint
- ☐ Thesis
- ☐ CR Release

NOTE. COMPLETE FOR PRESENTATIONS, JOURNAL PUBLICATION AND PREPRINTS

5. Name of organization or professional meeting, location and date held	6. Name of journal, proceedings, etc.
TMX-71366	

7. PUBLICATION CONTENT CONSIDERATIONS.

DOES	DOES NOT	
☐	☒	Contain Classified Material
☐	☒	Use International System of Units (SI)
☐*	☒	Describe Potentially Patentable Subject Matter (I.E., A new and useful process, product, mechanical and electrical arrangement of parts, or composition or matter.)

*Forward to Patent Counsel, Code 204

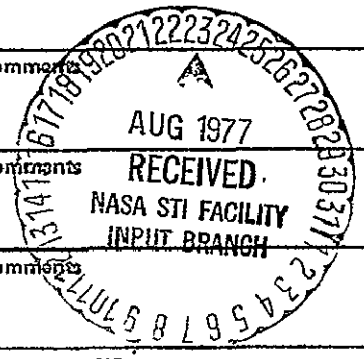
7a. Typed name and signature of reviewer in Office of Patent Counsel

8. This paper has been prepared keeping specifically in mind trade secrets or suggestions of outside individuals or concerns which have been communicated to the Center in confidence, and does not violate any such disclosures, and has been examined to see that all participating groups have been given proper credit.

Christine McQuillan
Primary Author

APPROVALS AND APPROVAL AUTHORITY (See instructions on the reverse of this form)

		Comments	Code	Date
9a Typed name and signature of Branch Head Paul H. Smith	☑ Approve ☐ Disapprove		626	7-21
b. Typed name and signature of Division Chief N. W. Spencer	☐ Approve ☐ Disapprove		620	7/25/77
c. Typed name and signature of Director of G. F. Pieper	☐ Approve ☐ Disapprove		600	
d. Typed name and signature Office of the Director	☐ Approve ☐ Disclaimer Waived ☐ Disapprove			



DOCUMENT RELEASE INSTRUCTIONS

10

A PREPRINTS & CONTRACT REPORTS ONLY

☒ Announce in STAR (no limitations on availability) ☐ Release by Originating Office Only

B WORKING PAPERS (Operational Documents) & CONTRACT REPORTS

1. Announce in CSTAR (distribution limited as indicated) 2 ☐ Release by Originating Office Only
   - a. ☐ U. S. Government Agencies and Contractors Only
   - b. ☐ U. S. Government Agencies Only
   - c. ☐ NASA and NASA contractors Only

11 **PRINTING SPECIFICATIONS (X-Documents Only)**

Normal printing includes the use of a standard 8 x 10½ cover, pages printed on both sides, with binding accomplished using two staples in the left margin.

Requester's Name	Job Order Number
Christine McQuillan	620-992-66-01-01

Code	Bldg	Room	Phone	Date of Request	Date Required	No. of Originals	No. of Copies
626	21	269	4321	6/23/77		88	50

GSFC 25-42 (8/74)